浙江大学

硕 士 学 位 论 文

论文题目：代数多重网格法研究及其在预处理 Krylov 子空间方法中的应用

作者姓名 陈书浩

指导教师 吴庆标 教授

学科(专业) 计算数学

所在学院 理学院

提交日期 2008 年 5 月

Research on Algebraic Multigrid Method and Application to several preconditioned Krylov subsapce methods

Candidate: Shuhao Chen

Supervisor: Professor Qingbiao Wu

Graduate School of Zhejiang University
Hangzhou,P.R.China
May,2008

摘 要

本文对代数多重网格法（**AMG**）在求解由偏微分方程有限元离散得到的大型稀疏对称线性方程组上的应用进行了研究。详细的介绍了代数多重网格算法，并给出了在算法具体实现时的一些方法和策略。对实际中碰到的几种网格（二维三角网格，三棱柱网格，四面体网格）给出了各自的粗化算法。并把代数多重网格算法和 **Krylov** 子空间法结合起来,推导出以 **AMG** 为预处理算子的预处理拟极小残差法。

本文也对 **Krylov** 子空间法及其预处理方法进行了描述，介绍了双共轭梯度法 **BiCG**（BiConjugate Gradient）和拟最小残差法 **QMR** （Quasi-Minimal Residual method）法，然后给出了它们的预处理方法。

最后，我们对一些求解实例给出了网格粗化结果，并对不同迭代算法进行了比较，相对其他方法，以 **AMG** 为预处理算子的 **QMR** 算法有较好的收敛效果。

关键词：代数多重网格法；Krylov 子空间法；AMG 预处理 QMR 算法

Abstract

In this thesis the algebraic multigrid method is considered for solving system equations which arise from a finite element discretization of partial differential equations. First, we introduce the algebraic multigrid. Then the coarsening strategies for different meshes which we encounter in practice are presented in this paper. The algebraic mulitigrid is used as a preconditioner in several preconditioned Krylov subspace methods to solve the system equations.

In this thesis, the Krylov subspace methods are introduced, such as BiConjugate Gradient method and Quasi-Minimal Residual method. Preconditioned BiConjugate Gradient method and preconditioned Quasi-Minimal Residual method are derived from the BiConjugate Gradient method and Quasi-Minimal Residual method.

Finally , numerical studies are given for solving system equations which arise from a finite element dicretization of partial differential equations. The coarsened meshes are given in figures. Several solvers are applied and their convergence behaviors are compared. AMG preconditioned methods have higher convergence than other methods.

Keywords: Algebraic Multigrid Method, Krylov Subspace Methods, Preconditioned Krylov Subspace Methods(which use AMG as the preconditioner)

目 录

第1章 绪论

在科学与工程计算领域，如流体力学计算，结构与非结构问题的有限元分析，石油地震数据处理，数值天气预报，电力系统优化设计，高维微分方程组数值解中，线性代数方程组的求解，特别是稀疏线性代数方程组的求解处于核心地位，大量实践经验表明，线性代数方程组的求解时间在整个问题的总计算时间中占有非常大的比重。由于稀疏矩阵包含着大量的零元素，需要设计专门的存储格式和算法，才能达到高效求解此类方程组的目的.

由于受舍入误差，计算机内存和计算复杂度的限制，对大规模问题，直接求解该类方程组几乎是不可能，通常采用迭代方法，迭代法的主要思想是通过构造有效的迭代格式，在有限步数内收敛于方程的精确解。在数值计算领域，人们一直在寻求具有最优计算复杂度的算法，即算法的计算量线性的以来于物理量个数，代数多重网格（**AMG**）方法[1,2]就具有这种性质，现已成为数值计算领域一种重要的加速迭代收敛技术。

多重网格法是在离散区域有限元网格的基础上构造出多个层次的网格，这些网格随着层数的增加越来越粗(点和边的数量越来越小)。线性方程组经过迭代光滑后，其高频部分误差会很明显地被削减，但低频部分误差很难被削减，但是当我们把迭代光滑后的误差通过一定机制传递到下一层网格上后，又变成既包含高频误差又包含低频误差,因此可以在下一层上继续光滑迭代,再把误差传递到更下一层网格……这样一直到最后一层网格，由于规模已经很小，就可以用直接法求解，然后将结果返回到上一层网格，作为该层解的修正，再作光滑迭代，把结果返回到更上一层网格……直到第一层网格，解修正后再作光滑迭代，我们就得到了线性方程组的近似解，其结果误差会比经典迭代法求得的误差小很多。

在实际应用中，多重网格法不仅直接用来求解线性方程组，而且作为预处理迭代法的预处理步骤更是个理想的选择，可以达到很好的收敛速度，可用来求解大规模线性方程组。

多重网格法根据网格层次的构造方法又分为几何多重网格法（GMG）和代数多重网格法(AMG)，最开始提出的是GMG，后来发展出AMG，AMG将网格层次的构造完全数值化，通过引入辅助矩阵来代表虚拟的网格，替代了GMG的实际网格，极大地扩大了多重网格法的适用范围

本论文第二章介绍了求解大型稀疏线性方程组的 **Krylov** 子空间法及其预处理方法，第三章介绍了通用的代数多重网格法并给出了一些在算法实现时的具体方法和策略，第四章给出了几种实际应用中常用网格的粗化方法，第五章提出了以代数多重网格法为预处理算子的预处理方法，推导出以 **AMG** 为预处理算子的预处理拟极小残差法，第六章给出了数值算例。

第 2 章 krylov 子空间法及其预处理方法

求解稀疏线形方程组的方法有直接法和迭代法二种，直接法中，由于要对稀疏矩阵进行分解，而在分解过程中将引入许多填充元素（即非零元素），从而增加存储开销，为了减少这种开销，需要在分解过程中，对矩阵的行和列进行重新的排序（即对矩阵进行置换），这就必须为减少填充元付出一定的代价，相对直接法，迭代法的存储开销大大介绍，一般地，每步迭代的计算开销与矩阵本身所需的存储开销同量阶，但迭代法的收敛性依赖于矩阵本身的特性，对某些条件不好的问题，迭代法可能不收敛，所以要对矩阵特性进行分析，不同问题需要采用不同的迭代方法。目前，Krylov 子空间方法是求解稀疏线性方程组最流行和最有效的方法之有一，也是当前研究的热点。

本章将先介绍 Krylov 子空间法及其方法分类，特别介绍双共轭梯度 BICG (Biconjugate Gradient) 法和拟最小残差法 QMR （Quasi-Minimal Residual method）法。然后介绍预处理方法的思想并推导出适用于 Krylov 子空间方法的预处理方法。

2.1 Krylov 子空间法

Krolov 子空间起主要思想是为各迭代步递归的构造残差向量，即第 n 步的残差向量，即第 n 步的残差向量 r_n 通过系数矩阵 A 的某个多项式与第一个残差向量 r_1 相乘得到到：

$$r_n = P_{m-1}(A)r_1$$

通常，迭代多项式的选取应使构造的残差向量在某种内积意义下相互正交，从而保证某种极小性（极小残差性），达到快速收敛的目的。

2.1.1 Krylov 子空间法概念及其方法的分类

给定线性方程组

$$Ax = b \qquad (2.1.1)$$

其中 $A \in R^{n\times n}, x, b \in R^n$。设 K_m 为 m 维子空间，一般投影方法是从 m 维仿射子空间 $X^{(0)} + K_m$ 中寻找（2.1.1）式的近似解 $X^{(m)}$，使相应的残差满足 **Petrov-Galerkin** 条件：

$$r_m = b - Ax^{(m)} \perp L_m$$

其中 L_m 为另一个 m 维子空间，$X^{(0)}$ 为迭代初值。如果 $K_m = K_m(A, r_0)$，则称 K_m 为 **Krolov** 子空间，而上述投影方法称为 **Krolov** 子空间，其中 r_0 为初始残差，$K_m(A, r_0)$ 定义为：

$$K_m(A, r_0) = span\left\{r_0, Ar_0, A^2 r_0, \cdots, A^{m-1} r_0\right\}$$

通常，针对（2.1.1）式设计的 **Krolov** 子空间方法具有如下二个特征：

（1） 极小残差性（或极小误差性），以保证收敛速度快；

（2） 每一步迭代的计算量与存储量较少，以保证计算的高效性。

下面介绍下 **Krolov** 子空间方法的分类，根据 K_m 和 L_m 的不同选取，可得到不同类型的 **Krolov** 子空间方法，主要可分为以下四类：

（1） 正交投影方法

取 $L_m = K_m(A, r_0)$，则这里 **Krolov** 子空间方法称为正交投影法，Hestenes 和 Stiefe[ ] 提出的共轭梯度法（**CG**）法是其中重要的方法，此时，需要 A 为对称正定矩阵。该法使 $d_m = X - X^{(m)}$ 在子空间 $X^{(0)} + K_m$ 中的能量范数达到极小，而且具有短的迭代计算公式，从而保证了计算的高效性。此后该法由 Cencus[3]发展成预条件 CG 方法。目前与各种预备条件子相结合的方法是求解大型对称正定稀疏线性方程组的重要的方法。

（2） 正交化方法

取 $L_m = AK_m(A, r_0)$，则这类 **Krolov** 子空间方法称为正交化方法，Saad[4]提出的

广义极小残差法（**GMRES**）法是这类方法中的典型代表，由于该类方法具有良好的数值稳定性，因此一直是人们研究的热点，并产生了变形方法[5，6]。

（3） 双正交化方法

取 $L_m = K_m(A^T, r_0)$，则这类 **Krolov** 子空间方法称为双正交方法。显然，当 A 为非对称的情形。Lanczos[7]提出的双共轭梯度法（**BICG**）法是其中最基本的方法。**BICG** 法在计算中也存在一些缺陷，许多研究者在 **BICG** 法的基础上，发展了一系列收敛性好的方法，如共轭梯度平方法（**CGS**）法[8]，广义共轭梯度平方法（**GCGS**）[9]，共轭梯度稳定性方法（**BICGSTAG**）法[10]，拟最小残差法（**QMR**）[11]法。

（4） 法方程组方法

取 $L_m = K_m(A^T A, A^T r_0)$，则这类 **Krolov** 子空间方法称为法方程组方法。这类方法是将 CG 法应用于法方程组 $A^T Ax = A^T b$ 或者 $A^T Au = Ab, x = A^T u$ 上，**CGNR** 和 **CGNE** 方法是这类方法的典型代表。

2.1.2 BICG 法和 QMR 算法

下面先介绍下常用的用于对称矩阵的 BICG 算法和由 BICG 算法改进来的适应复系数矩阵的 BICGSTAB 算法，BICGSTAB 算法比 BICG 算法具有更好的收敛速度，且不要计算矩阵的共轭转置（具体描述见[12]）。算法见 2.2.3 和 2.2.4。

给定处值x

$r=b-Ax$

$p=r$

$\rho=\mathrm{Re}(r,\bar{r})$

for $k=1,2,...KMAX$

$w=Ap$

$\alpha=\rho/\mathrm{Re}(w,\bar{r})$

$x=x+\alpha p$

$r=r-\alpha w$

if $\|r\|/\|b\|\le\varepsilon$

break;

end

$\bar{\rho}=\rho$

$\rho=\mathrm{Re}(r,\bar{r})$

$\beta=\rho/\rho$

$p=r+\beta p$

end

算法2.2.3对称矩阵的BiCG法

给定处值x

$r=b-Ax$

$\rho=\alpha=w=\tilde{\rho}=1$

$v=p=0$

$r_0=r$

for $k=1,2,...KMAX$

$\rho=(r_0,r)$

$\beta=\frac{\rho}{\tilde{\rho}}\cdot\frac{\alpha}{w}$

$\tilde{\rho}=\rho$

$p=r+\beta(p-w\cdot v)$

$v=Ap$

$\alpha=\rho/(r_0,r)$

$s=r-\alpha v$

$t=As$

$w=(t,s)/(t,t)$

$x=x+\alpha p+ws$

$r=s-wt$

if $\|r\|/\|b\|\le\varepsilon$

break;

end

end

算法2.2.4 BICGSTAB法

BICG算法也存在一些缺陷，比如会产生不规则的收敛行为（残差范数强烈振荡），出现除比较接近0的数（导致下面的迭代不稳定）的情况。在BICG基础上改进产生的CGS[8]和BICGSTAB[10]虽然在一些问题比BICG效率高些，但是它们没解决BICG可能不收敛的的问题，仍然和BICG 一样存在不稳定性。后来RW.Freund提出了 QMR 算法[11]。QMR算法通过其残差的特性克服了BICG的收敛不光滑的特点，是在BICG的基础上产生的一种收敛性比较好的算法。下面介绍下QMR算法，见算法2.2.5。

给点初值 x_0

$r_0 = b - Ax_0, \rho_0 = \| r_0 \|, v_1 = r_0 / \rho_0$

$V_1 = v_1, W_1 = v_1, D_1 = W_1^T V_1, \rho_1 = \xi_1 = 1$

$d_0 = 0, d_1 = V_1 D_1^{-1} W_1^T, f_0 = 0, f_1 = W_1 D_1^{-T} V_1^T$

for $n = 1, 2, \ldots$

$\widetilde{v_{n+1}} = Av_n - d_1 Av_n - d_0 Av_n$

$\widetilde{w_{n+1}} = A^T w_n - f_1 A^T w_n - f_0 A^T w_n$

$\rho_{n+1} = \| \widetilde{v_{n+1}} \| \quad \xi_{n+1} = \| \widetilde{w_{n+1}} \|$

if $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$ *break*

end

$v_{n+1} = \widetilde{v_{n+1}} / \rho_{n+1} \qquad w_{n+1} = \widetilde{w_{n+1}} / \xi_{n+1}$

$V_{n+1} = [V_n \; v_{n+1}] \quad W_{n+1} = [W_n \; w_{n+1}] \quad D_{n+1} = W_{n+1}^T V_{n+1}$

$d_0 = d_1 \quad d_1 = V_{n+1} D_{n+1}^{-1} W_{n+1}^T \quad f_0 = f_1 \quad f_1 = W_{n+1} D_{n+1}^{-T} V_{n+1}^T$

得出 $AV_n = V_n H_n + [0 \ldots 0 \; \widetilde{v_{n+1}}]$ 中的 H_n

$$H_n^{'} = \begin{bmatrix} H_n \\ \rho_{n+1} (e_n)^T \end{bmatrix}, e_n = [0, \ldots 0 \; 1]^T \in \mathbb{R}^n,$$

对 $H_n^{'}$ 进行 QR 分解得 $Q_n H_n^{'} = \begin{bmatrix} R_n \\ 0 \end{bmatrix}$

$t_n = [I_n \;\; 0] Q_n d_n$ 其中 $d_n = \rho_0 e_{n+1} \quad e_{n+1} = [0, 0 \ldots 1] \in \mathbb{R}^{n+1}$

$x_n = x_0 + V_n R_n^{-1} t_n$

end

算法 2.2.5

注：一些实现细节可见文献[11]。

2.2 预处理方法

当线性方程组的系数矩阵条件不够好时，我们可以通过预处理方法把系数矩阵

转化为一个系数矩阵只有少数几个互不相同的特征值或非常良态的等价方程组，然后对等价方程组使用迭代法求解。对（2.1.1）预处理方法的一般转化模式：令 $\tilde{A}=M_1^{-1}AM_2^{-1},\tilde{b}=M_1^{-1}b,\tilde{x}=M_2x$ 则有与（2.1.1）等价方程组

$$\tilde{A}\tilde{x}=\tilde{b} \qquad (2.3.1)$$

$M=M_1M_2$ 称为预优矩阵。预处理方法成功的关键在于预优矩阵 M 是否选择合适。一个好的预优矩阵应该具有如下特征：（1）M 是对称正定的，（2）M 是稀疏的，（3）$M^{-1}A$ 仅有少数几个互不相同的特征值或者其特征值集中在某点附近，（4）形如 $Mz=r$ 方程组易于求解。当然对一般的线性方程组要选择一个同时满

足上述 4 个条件的预优矩阵 M 往往是十分困难的，下面是几种常用的选取方式：

对角预优矩阵,不完全 Cholesky 因子预优矩阵，多项式预优矩阵，详细见文献[13]。

下面介绍下预处理 **BICGSTAB** 法和预处理 **QMR** 法，详细见文献[14]。

预处理 BICGSTAB 法

令 $M=C^2$，用 $C^{-1}AC,Cx,C^{-1}b,C^{-1}r,Cp,Cv,Cs,C^{-1}t$，替换算法 2.2.4 中的 A,x,b,r,p,v,s,t 整理后得到预处理算法 **BICGSTAB** 法，见算法 2.3.1。然后在 2.3.1 算法基础上用 r 代替 $M^{-1}r$，得到实用的预处理 **BICGSTAB** 法见算法 2.3.2。

给定初值 x

$r = b - Ax$

$\rho = \alpha = w = \widetilde{\rho} = 1$

$v = p = 0$

$r_0 = r$

$for \quad k = 1, 2, \ldots KMAX$

$\rho = (r_0, M^{-1}r)$

$\beta = \frac{\rho}{\widetilde{\rho}} \cdot \frac{\alpha}{w}$

$\widetilde{\rho} = \rho$

$p = M^{-1}r + \beta(p - w \cdot v)$

$v = M^{-1}AP$

$\alpha = \rho / (r_0, r)$

$s = M^{-1}r - \alpha v$

$t = As$

$w = (t, s) / (t, M^{-1}t)$

$x = x + \alpha p + ws$

$M^{-1}r = s - wM^{-1}t$

$if \quad \| r \| / \| b \| \leq \varepsilon$

$break;$

end

end

算法 2.3.1

给定初值 x

$r = b - Ax$

$r = M^{-1}r_0$

$\rho = \alpha = w = \widetilde{\rho} = 1$

$v = p = 0$

$for \quad k = 1, 2, \ldots KMAX$

$\rho = (r_0, r)$

$\beta = \frac{\rho}{\widetilde{\rho}} \cdot \frac{\alpha}{w}$

$\widetilde{\rho} = \rho$

$p = r + \beta(p - w \cdot v)$

求解 $Mv = Ap$的 v

$\alpha = \rho / (r_0, r)$

$s = r - \alpha v$

$t = As$

求解 $Mz = t$的 z

$w = (t, s) / (t, z)$

$x = x + \alpha p + ws$

$r = s - wz$

$if \quad \| Ax - b \| / \| b \| \leq \varepsilon$

$break;$

end

end

算法 2.3.2

说明：我们在把预处理前的算法改成预处理后的算法时，也对其中变量进行替换如上述算法中把 Cs, Cp, Cv 代替 s, p, v ，以尽量减少新算法中 M, M_1, M_2 这些的次数避免算法增加过多计算量。

预处理 QMR 法

在算法 2.2.5 中用 $M^{-1}A, M^{-1}b$ 代替 A, b，得预处理 **QMR** 算法，见算法 2.3.3。

给点初值 x_0

$r_0 = b - Ax_0, r = M^{-1}(b - Ax_0), \rho_0 = \| r \|, v_1 = r / \rho_0$

$V_1 = v_1, W_1 = v_1, D_1 = W_1^T V_1, \rho_1 = \xi_1 = 1$

$d_0 = 0, d_1 = V_1 D_1^{-1} W_1^T, f_0 = 0, f_1 = W_1 D_1^{-T} V_1^T$

for $n = 1, 2, \ldots$

$Mz = Av_n$

$\widetilde{v_{n+1}} = z - d_1 z - d_0 z$

$\widetilde{w_{n+1}} = A^T w_n - f_1 A^T w_n - f_0 A^T w_n$

$\rho_{n+1} = \| \widetilde{v_{n+1}} \| \quad \xi_{n+1} = \| \widetilde{w_{n+1}} \|$

if $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$ *break*

end

$v_{n+1} = \widetilde{v_{n+1}} / \rho_{n+1} \qquad w_{n+1} = \widetilde{w_{n+1}} / \xi_{n+1}$

$V_{n+1} = [V_n \; v_{n+1}] \quad W_{n+1} = [W_n \; w_{n+1}] \quad D_{n+1} = W_{n+1}^T V_{n+1}$

$d_0 = d_1 \quad d_1 = V_{n+1} D_{n+1}^{-1} W_{n+1}^T \quad f_0 = f_1 \quad f_1 = W_{n+1} D_{n+1}^{-T} V_{n+1}^T$

得出 $AV_n = V_n H_n + [0 \ldots 0 \; \widetilde{v_{n+1}}]$ 中的 H_n

$$H_n^{'} = \begin{bmatrix} H_n \\ \rho_{n+1}(e_n)^T \end{bmatrix}, e_n = [0, \ldots 0 \; 1]^T \in \mathbb{R}^n,$$

对 $H_n^{'}$ 进行 QR 分解得 $Q_n H_n^{'} = \begin{bmatrix} R_n \\ 0 \end{bmatrix}$

$t_n = [I_n \;\; 0] Q_n d_n$ 其中 $d_n = \rho_0 e_{n+1} \quad e_{n+1} = [0, 0 \ldots 1] \in \mathbb{R}^{n+1}$

$x_n = x_0 + V_n R_n^{-1} t_n$

end

算法 2.3.3

在预处理迭代方法中需要求解 $MX = Av$ 的方程，这里就可以与代数多重网格结合起来，产生以代数多重网格法为预处理步骤的 **Krylov** 子空间法。

第 3 章 代数多重网格法

多重网格法是在离散区域有限元网格的基础上构造出多个层次的网格，这些网格随着层数的增加越来越粗(点和边的数量越来越小)。线性方程组经过迭代光滑后，其高频部分误差会很明显地被削减，但低频部分误差很难被削减，但是当我们把迭代光滑后的误差通过一定机制传递到下一层网格上后，又变成既包含高频误差又包含低频误差，因此可以在下一层上继续光滑迭代，再把误差传递到更下一层网格……这样一直到最后一层网格，由于规模已经很小，就可以用直接法求解，然后将结果返回到上一层网格，作为该层解的修正，再作光滑迭代，把结果返回到更上一层网格……直到第一层网格，解修正后再作光滑迭代，我们就得到了线性方程组的近似解，其结果误差会比经典迭代法求得的误差小很多。

多重网格法根据网格层次的构造方法又分为几何多重网格法（**GMG**）和代数多重网格法(**AMG**)，最开始提出的是**GMG**，后来发展出**AMG**。几何多重网格法需要提供多层有限元网格而这是经常做不到的，另一个方面几何多重网格得到的最粗网格的线性方程组规模仍然太大，用直接法或者迭代法仍然不能有效的求解。而**AMG**将网格层次的构造完全数值化，通过引入辅助矩阵来代表虚拟的网格，替代了**GMG**的实际网格，极大地扩大了多重网格法的适用范围。

本章将介绍求解由边型有限元法(Edge Finite Element)离散得到的线性方程组的代数多重网格的一般方法。代数多重网格法分为代数多重网格的构造和代数多重网格求解两个部分。

3.1 代数多重网格的构造

代数多重网格的构造是代数多重网格法的核心部分，它负责生成多重虚拟有限元网格（即下文中提到的辅助矩阵）、各层网格上的线性方程组（即系数矩阵）和两个相邻网格层之间的传递矩阵。在代数多重网格的构造已知的是：系数矩阵 A 和有限元网格 $w=(w^n,w^e)$，w^n 是点集，w^e 是边集。

第一层的构造

有限元网格 $w=(w^n,w^e)$ 作为第一层有限元网格 $w_1=(w_1^n,w_1^e)$，根据 w_1^n 构造辅助矩阵 B_1，将系数矩阵 A 作为第一层辅助矩阵 A_1。

对 $l=1,2...$

第 $l+1$ 层的构造

根据第 l 层辅助矩阵 B_l 粗化得到第 $l+1$ 层粗网格节点 w_{l+1}^n，以及第 l 层节点 w_l^n 到第 $l+1$ 层节点 w_{l+1}^n 的传递矩阵 P_l^n，构造辅助矩阵 $B_{l+1}=(P_l^n)^T B_l P_l^n$，根据辅助矩阵构造第 l 层边 w_l^e 到第 $l+1$ 层边 w_{l+1}^e 的传递矩阵 P_l^e，构造辅助矩阵 $A_{l+1}=(P_l^e)^T A_{l+1} P_l^e$。

具体算法描述见算法3. 1。

Setup(A_l,B_l,l)
// *CoarseGridSize*在函数调用前设置
if $|w_l^e| \le$ *CoarseGridSize then*
    return;
else
    将w_l^n分成C 和 F
    令$w_{l+1}^n=C$
    构造辅助矩阵的传递算子P_l^n
    计算粗网格上辅助矩阵$B_{l+1}=(P_l^n)^T B_l P_l^n$
    构造系数矩阵的传递算子P_l^e
    计算粗网格上的系数矩阵$A_{l+1}=(P_l^e)^T A_{l+1} P_l^e$
    Setup$(A_{l+1},B_{l+1},l+1)$
end

算法3.1 边型有限元代数多重网格构造算法

说明：

在算法中辅助矩阵 B_l 可以提供足够的网格信息，将 w_l^n 分为 C 和 F 实际上不是对 w_l^n 进行操作，而是在辅助矩阵上进行判断和操作。

3.1.1 辅助矩阵

系数矩阵 A 是在有限元网格 $w=(w^n,w^e)$ 上通过有限远法离散得到的，其中 w^n 是点集，$|w^n|=N^n$ 为点的个数，w^e 为边的集合，$|w^e|=N^e$ 为边的条数。这里的边是指一条几何边的两个顶点的索引构成的整数对。比如设 $i,j\in w^n$ 分别是顶点 $x_i,x_j\in\mathbb{R}^d$ 的索引（其中 $d=2$ 表示二维顶点，$d=3$ 表示三维顶点）。那么边为 $e_{ij}=(i,j)\in w^e$ 见图3. 1。

i,j 为网格点，(i,j) 为连接 i,j 的边

我们将由网格信息构造符合如下条件的辅助矩阵 $B \in \mathbb{R}^{N^n \times N^n}$：

$$B = \begin{cases} b_{ij} \le 0 & i \ne j \\ -\sum_{j \ne i} b_{ij} > 0 & i = j \end{cases} \qquad (3.1)$$

满足（3.1）的矩阵叫做 Z 矩阵，这样可以保证：

a） b_{ij} 可以反映两个集合点 i, j 之间的距离；

b） $B_1 = B$ 是 Z 矩阵，则通过 $B_{l+1} = (P_l^n)^T B_l P_l^n$ 得到的辅助矩阵多是 Z 矩阵。

b_{ij} 的定义可以采取不同的策略，一般采用下面两种方法中的一种：

a) B反映几何有限元网格，即 $b_{ij} \ne 0 \Leftrightarrow (i, j)$ 是有限元网格上的一条边。比如可去：

$$b_{ij} = -\frac{1}{\| x_i - x_j \|^2}, i \ne j$$

其中 $x_i, x_j \in \mathbb{R}^d (d = 2,3)$ 为顶点。

b） B 和矩阵 A 采取一样的样式，即 $\| A_{ij} \|_\infty \ne 0 \Leftrightarrow | b_{ij} | \ne 0$，比如可取

$$b_{ij} = -\| A_{ij} \|_\infty, i \ne j$$

说明：

只有第一层网格上的辅助矩阵需要根据网格几何信息来构造。在定义了适当的传递算子后，其它各层粗网格上的辅助矩阵可以直接通过**Galerkin**方法计算得到，而不需要知道粗网格的几何结构。这体现了代数多重网格法把对网格的操作完全数值化。

辅助矩阵代表了虚拟的有限元网格。它需具有如下性质：

1） 如果一个点 j 离另一个点 i 较远，则辅助矩阵元素 b_{ij} 是一个较小的负值；

2） 如果一个点 j 离另一个点 i 较远，则辅助矩阵元素 b_{ij} 是一个较大的负值；

3.1.2 网格的粗化

辅助矩阵 B 代表了虚拟的有限元网格 $w=(w^n,w^e)$，我们将它划分为两个不相交的点集，即

$$w^n = w_C^n \bigcup w_F^n, w_C^n \bigcap w_F^n = \varnothing$$

w_C^n 代表粗点集合，w_F^n 代表细点集合。

在介绍粗化算法前，我们先引入下列集合定义：

$$\begin{aligned}
&N^i = \{j \in w^n : |b_{ij}| \neq 0, i \neq j\},\\
&S^i = \{j \in N^i : |b_{ij}| > coarse(B,i,j), i \neq j\},\\
&S^{i,T} = \{j \in N^i : i \in S^j\},\\
&C^i = N^i \bigcap w_C^n,\\
&F^i = N^i \bigcap w_F^n
\end{aligned}$$

其中 $N^i, S^i, S^{i,T}$ 分别是节点 i 的邻接点集合，强连接点集合，强连接到 i 点集合。

截断函数 $coarse(B,i,j)$ 几种取法如下：

$$coarse(B,i,j) = \begin{cases} \theta\sqrt{|b_{ii}||b_{jj}|} \\ \theta \max_{l \neq i} |b_{il}| \quad \theta \in (0,1) \\ \theta \end{cases}$$

可以看出当截断函数取第 1，3 项时 $S^i = S^{i,T}$，取第 2 项时 $S^i \neq S^{i,T}$。

粗网格的构造需遵循下面两个准则：

C1）对每个 $i \in w_C^n, j \in S^i$，要么属于 w_C^n，要么至少强连接到 C^i 的一个点。

C2）w_C^n 是其中任何两点多不形成强连接关系的最大集合。

下面介绍 RS 粗化算法[15]：

算法主要分为 2 个阶段

第一阶段，快速选取尽可能多的 点，以尽量满足（C2）

起初，选取 $S^{i,T}$ 最大的 i 作为 w_C^n 点，而 $S^{i,T}$ 置入 w_F^n 中。

之后，算法趋向于选取强影响更多的即 $|S^{i,T}|+|S^{i,T} \bigcap w_F^n|$ 最大的顶点为 w_c^n 点，这样可以减少 $w_C^n - w_C^n$ 连接，而且算法保证了每个 w_F^n 点至少强依赖于一个 w_C^n 点。具体

见算法 3.2.1 和算法 3.2.2。

$CoarseI(w^n, \{S^{i,T}\})$
$w_c^n \leftarrow \varnothing, w_F^n \leftarrow \varnothing$
while $w_c^n \bigcup w_F^n \neq w^n$ *do*
　选取$w^n \setminus \{w_c^n \bigcup w_F^n\}$中$|S^{i,T}|+|S^{i,T} \bigcap w_F^n|$最大的$i$
if $|S^{i,T}|+|S^{i,T} \bigcap w_F^n|=0$ *then*
　$F \leftarrow w^n \setminus w_c^n$
else
　$w_c^n \leftarrow w_c^n \bigcup \{i\}$
　$w_F^n \leftarrow w_F^n \bigcup \{S^{i,T} \setminus w_c^n\}$
end
end

算法 3.2.1　粗化算法第一阶段

$$
\begin{array}{l}
CoarseI(S,W) \\
C \leftarrow \varnothing, F \leftarrow \varnothing, U \leftarrow W \\
z_i \leftarrow |S_i^T|, i = 1,2,...,|W| \\
\quad while\ U \neq \varnothing\ do \\
\quad\quad pick\ i\ s.t.\ z_i = \max_{k \in U}\{z_k\} \\
\quad\quad C \leftarrow C \bigcup \{i\} \\
\quad\quad U \leftarrow U - \{i\} \\
\quad\quad for\ \ j \in S_i^T \bigcap U\ \ do \\
\quad\quad\quad F \leftarrow F \bigcup \{j\} \\
\quad\quad\quad U \leftarrow U - \{j\} \\
\quad\quad\quad for\ k \in S_j \bigcap U\ \ do \\
\quad\quad\quad\quad z_k \leftarrow z_k + 1 \\
\quad\quad\quad end\ \ for \\
\quad\quad end\ for \\
\quad\quad for\ j \in S_i^T \bigcap U\ do \\
\quad\quad z_j \leftarrow z_j - 1 \\
\quad\quad end\ \ for \\
end\ \ while
\end{array}
$$

算法 3.2.2　粗化第一阶段算法比 3.2.1 容易实现

第二阶段，对余下的结点进行检测，确保结点满足（C1），算法 3.2.3 为第二阶段算法的具体描述，其中 $d(i,I) = \left(\sum_{j \in I} -B_{ij}\right) \Big/ \max_{k \neq i}\{|B_{ik}|\}$。

$CoarseII(S,C,F)$

$T \leftarrow \varnothing$

while $F \setminus T \neq \varnothing$ *do*

pick $i \in F \setminus T$

$T \leftarrow T \cup \{i\}$

$\widetilde{C} \leftarrow S^i \cap C$

$C^i \leftarrow S^i \cap F$

while $F^i \neq \varnothing$ *do*

选取 $j \in F^i, F^i = F^i \setminus \{j\}$

if $d(j,C^i) \leq \beta d(i,\{j\})$ *then*

if $\widetilde{C} = \varnothing$ *then*

$\widetilde{C} \leftarrow \{j\}, C^i = C^i \cup \{j\}$

else

$C = C \cup \{i\}, F = F \setminus \{i\}$

end

end

$C \leftarrow C \cup \widetilde{C}, F \leftarrow F \setminus \widetilde{C}$

end

end

$w_c^n \leftarrow C$

$w_F^n \leftarrow F$

算法 3.2.3 粗化第二阶段算法，对细点集进行检测，以满足条件（C1），细点集中不满足（C1）条件的结点则将起放如粗点集中。

图 3.2 为网格粗化的一个示意图，可以帮助理解粗化算法。

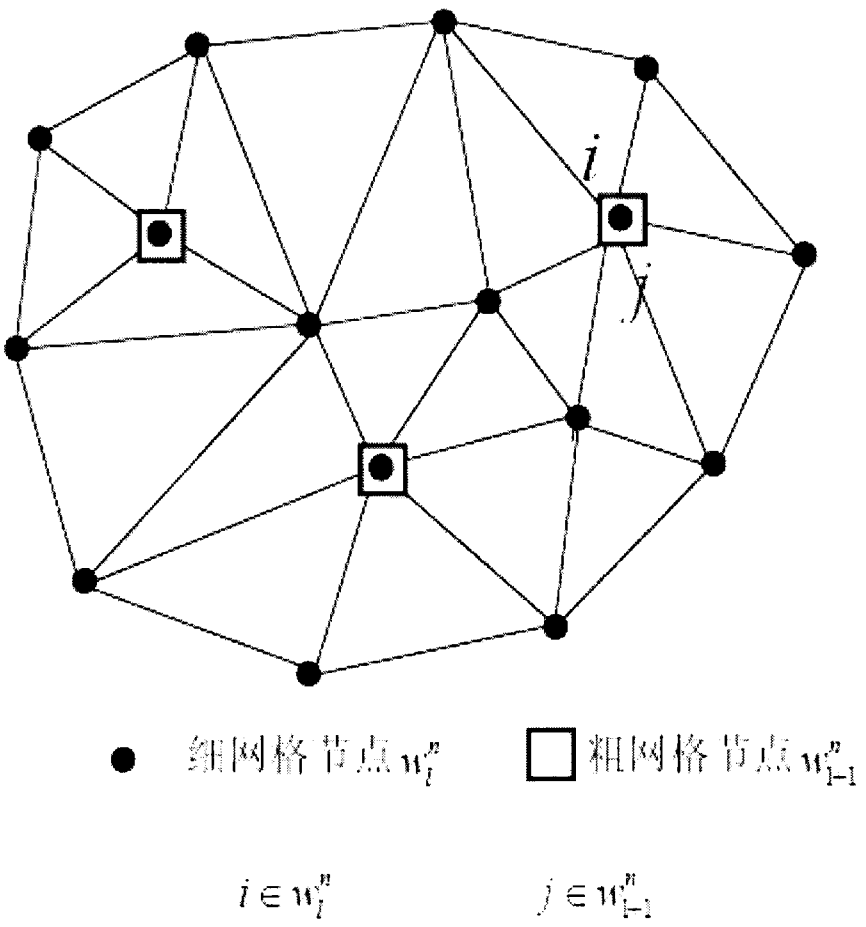


图 3.2 网格粗化示意图

同一个节点，在细网格中编号为i，但在粗网格中编号为j。

在对第 l 层粗化后，将 w_C^n 中点设为第 $l+1$ 层网格节点集，即 $w_{l+1}^n \leftarrow w_c^n$。

从上述算法可以看出：

1）实际粗化只需要对辅助矩阵 B 和其中元素进行各种操作，并不需要几何网格；

2）这里 w_C^n 中保存的是第 l 层网格上的节点索引（如图 3.2 中的 $i \in w_l^n$），而 w_{l+1}^n 保存的是第 $l+1$ 层网格上的节点索引（如图 3.2 中的 $j \in w_{l+1}^n$），因此 $w_{l+1}^n \leftarrow w_c^n$

后的网格节点会被赋于新的索引值。在后面内容中我们将通过例子来说明。

3.1.3 传递算子

在代数多重网格法中，粗网格上矩阵（包括辅助矩阵和系数矩阵）的构造完全取决于传递算子和当前网格的矩阵。第一层网格上辅助矩阵是 Z 矩阵，系数矩阵也有很好的性质（比如对称正定），因此我们必须对传递算子加一些限制条件，使得粗网格上矩阵同样具有当前网格矩阵的性质。下面介绍 Reitzinger 和 Schoberl 提出的的传递算子[16,17,18]。

细网格到粗网格映射的定义

在上一小节的最后我们提到，在对第 l 层有限元网格 $w_l=(w_l^n,w_l^e)$ 粗化后，将 w_C^n 中点设为第 $l+1$ 层网格节点集 w_{l+1}^n，节点会被赋予新的索引值，象图 3.2 的情况，同一个节点在 w_l^n 中的索引为 i，而在 w_{l+1}^n 中的索引为 j。因此，当我们对第 $l+1$ 层网格节点指定新的索引值后，就可以得到从 w_C^n 到 w_{l+1}^n 的一对一映射 $idx(\cdot)$：

$$idx: w_c^n \to w_{l+1}^n$$

或者

$$w_{l+1}^n = idx(w_c^n)$$

例如图 3.2 中，$j=idx(i)$。

我们将 w_l^n 分为互不相交的聚类 $(I^i)_{i=1}^{N_{l+1}^n}(|w_{l+1}^n|=N_{l+1}^n<N_l^n)$，其中

$$I^i \cap I^j = \varnothing, \bigcup_{i=1}^{N_{l+1}^n} I^i = w_l^n$$

图 3.3 为节点归类示意图。

这里涉及到如何归类，象图 3.3 中，既可以令 $r\in I^i$，也可以令 $r\in I^k$，这可以通过下面这个原则解决，如果 t 是细节点，采取与 t 邻接的所有粗节点中与 t 距离最近

的节点所在聚类做为t的聚类。比如图 3.3 中，r离i的距离比r离k的距离小，因此令$r\in I^i$。由图可知I^i是由第i个粗点和它旁边的部分细点组成的。

有了上面的归类，我们可以把从w_c^n到w_{l+1}^n的映射$idx(\cdot)$沿拓到从w_l^n到w_{l+1}^n：

$$idx: w_l^n \to w_{l+1}^n$$

或者

$$w_{l+1}^n = idx(w_l^n)$$

例如图 3.3 中，$i = idx(r), j = idx(s)$。

映射$idx(\cdot)$沿拓之后，聚类$(I^i)_{i=1}^{N_{l+1}^n}$也可以表示为

$$I^i = \{j \in w_l^n : idx(j) = i\}$$

粗网格节点集合可以表示为

$$w_{l+1}^n = \{idx(i) : i \in w_l^n\}$$

$r, s \in w_l^n, i, j, k \in w_{l+1}^n, r \in I^i, s \in I^j$

图 3.3 粗化后网格节点归类示意图 r离 i的距离比离 j的距离近，故$r\in I^i$

辅助矩阵传递算子

从第l层到第$l+1$层的辅助矩阵传递算子$P_l^n \in \mathbb{R}^{N_l^n \times N_{l+1}^n}(N_l^n = |w_l^n|, N_{l+1}^n = |w_{l+1}^n|)$所有元素为1或0，定义如下：

$$(P_l^n)_{ij} = \begin{cases} 1 & j = idx(i), i \in w_l^n, j \in w_{l+1}^n \\ 0 & else \end{cases} \tag{3.2}$$

粗网格辅助矩阵

根据Galerkin方法计算，即

$$B_{l+1} = (P_l^n)^T B_l P_l^n \tag{3.3}$$

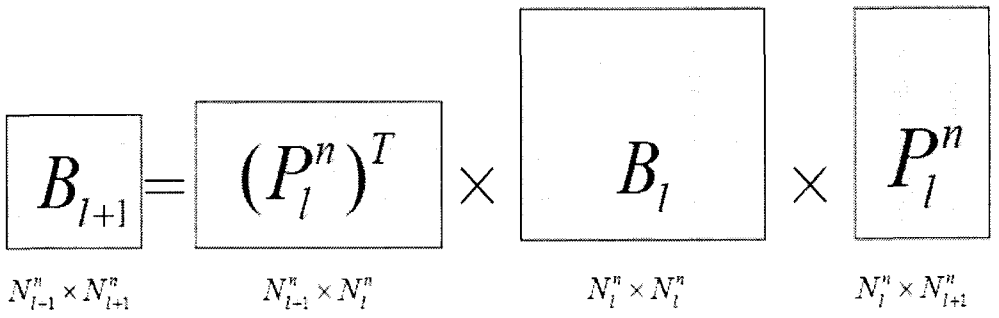


图3.4辅助矩阵的Galerkin方法示意图

通过此方法得到的B_{l+1}代表第$l+1$层虚拟有限元网格且B_{l+1}是Z矩阵。具体证明见文献[19]。

在计算得B_{l+1}后，我们就可以确定粗网格边w_{l+1}^e，因此得到了虚拟粗网格$w_{l+1} = (w_{l+1}^n, w_{l+1}^e)$。见图3.4。

说明：在粗网格中点的索引是根据 RS 粗化算法中粗点被选的先后顺序来排的。如先选出的第一个粗点指定索引值为 0，第二个粗点索引值为 1，粗网格的边索引可以根据点的索引以下列顺序排列如：（0，1）（0，2）(1，2)（1，3）…。

系数矩阵传递算子

从第 l 层到第 $l+1$ 层的系数矩阵传递算子 $P_l^e \in \mathbb{R}^{N_l^e \times N_{l+1}^e} (N_l^e = |w_l^e|, N_{l+1}^e = |w_{l+1}^e|)$, 对于细网格边 $i=(i_1,i_2) \in w_l^e, i_1 < i_2$ 和粗网格边 $j=(j_1,j_2) \in w_{l+1}^e, j_1 < j_2$， P_l^e 定义如下：

$$(P_l^e)_{ij} = \begin{cases} 1 & when \ (j_1,j_2) = (idx(i_1), idx(i_2)) \\ -1 & when \ (j_1,j_2) = (idx(i_2), idx(i_1)) \\ 0 & else \end{cases}$$

例：图 3.4 中，$(P_l^e)_{ij} = 1$。

粗网格系数矩阵

根据 **Galerkin** 方法计算，即

$$A_{l+1} = (P_l^e)^T A_l P_l^e \qquad (3.6)$$

对系数矩阵的传递算子的性质分析可见文献[18]。

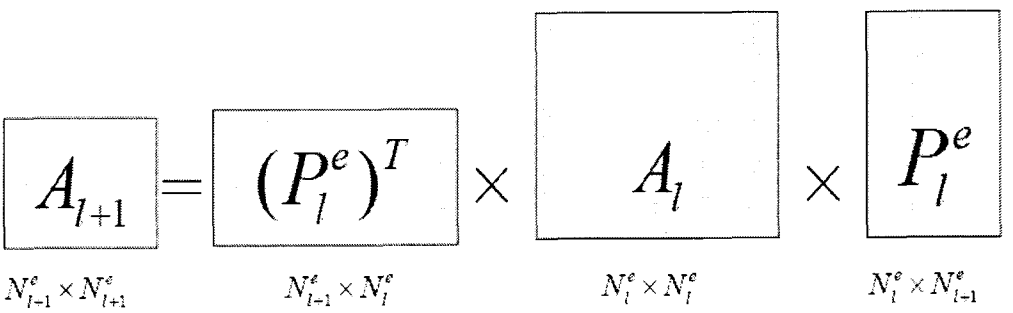


图 3.6 系数矩阵的 Galerkin 方法

3.2 代数多重网格法的求解

3.2.1 算法描述

在构造了多重网格后，我们就可以进行求解，代数多重网格的求解有 V 循环，W 循环等多种，这里介绍 V 循环，其他的算法见文献[20]。V 循环描述见算法 3.3。

$V(v_F, v_B)$循环：$AMGStep(x_l, b_l, A_l, P_l^e, l)$
if 最后一层 *then*
　　用直接法求解 $x_l = (A_l)^{-1} b_l$
else
　　对$A_l x_l = b_l$前向光滑迭代v_F次
　　计算差值$\mathrm{d}_l = b_l - A_l x_l$
　　将差值传递到下一层$\mathrm{d}_{l+1} = (P_l^e)^T d_l$
　　设置$x_{l+1} = 0$
　　递归调用　$AMGStep(x_{l+1}, d_{l+1}, A_{l+1}, P_{l+1}^e, l)$
　　将修正值返回到上一层 $\mathrm{s}_l = P_l^e x_{l+1}$
　　更新解 $x_l = x_l + s_l$
　　对$A_l x_l = b_l$后向光滑迭代v_B次
end

算法 3.3 V 循环代数多重网格求解算法

3.2.2 光滑迭代算子

对于系数矩阵是实系数对称正定（**SPD**）的情况，经典的迭代法多可以作为光滑迭代算子，比如 **Jacobi** 迭代，**Gauss-Seidel** 迭代等。也可采用 **Arnold** 光滑算子（见文献[ ]），以及 **Hiptmair** 光滑算子（见文献[21]）。

对于系数矩阵是复系数的情况，可以采用复数版本的 **Arnold** 光滑算子（见文献[ ]），如果矩阵 **Gauss-Seidel** 迭代收敛，当然也可采用 **Gauss-Seidel** 迭代。

Gauss-Seidel 的前向和后向迭代见算法 3.4 和算法 3.5 。

给定初值 $x^{(0)}$	给定初值 $x^{(0)}$
$for\ k=1,2,...$	$for\ k=1,2,...$
$for\ i=1,2,...n$	$for\ i=n,n-1,...1$
$\sigma=0$	$\sigma=0$
$for\ \ j=1,2,...,i-1$	$for\ \ j=1,2,...,i-1$
$\sigma=\sigma+a_{ij}x_j^{(k)}$	$\sigma=\sigma+a_{ij}x_j^{(k)}$
end	end
$x_i^{(k)}=(b_i-\sigma)/a_{ii}$	$x_i^{(k)}=(b_i-\sigma)/a_{ii}$
end	end

算法 3.4**Gauss-Seidel** 前向迭代　算法 3.5**Gauss-Seidel** 后向迭代

3.2.3 直接解法的选取

由于最后一层的线性方程组的规模最小（比如设定为未知个数为 500 时不继续粗化网格），因此一般的直接解法多适用，而且求解速度多很快。

3.2.4 收敛性说明

文献[22],[23]中给出了特定的多重网格法的收敛性分析，文献[24]给出了**Petrov-Galerkin**代数多重网格法的收敛性分析，具体见相关文献。

虽然无法在理论上给出代数多重网格法收敛性普遍性的证明，但大量的数值实验显示代数多重网格法的收敛速度可以和几何多重网格法相媲美。代数多重网格法在各个工业领域里的应用可见文献[25,26,27,28]。

3.3 代数多重网格算法（V 循环）小结

代数多重网格算法的整体过程如下：

(一)代数多重网格的构造过程：

（1） 从有限元网格 $w=(w^n,w^e)$ 和初始条件得到辅助矩阵 B_1（虚拟网格）和系数矩阵 A_1；

（2） 利用 RS 粗化算法得出各层的辅助矩阵 $B_1...B_n$，各层的系数矩阵 $A_1...A_n$，各层之间的系数矩阵传递算子 $P_1^e...P_n^e$，直到最粗网格的系数矩阵个数小于指定的值（即可用直接法简单求解）。

（二）代数多重网格的求解（V 循环）

（1） 从第一层网格到最粗层网格逐层进行前光滑；

（2） 然后对最粗层的系数矩阵精确求解；

（3） 从最粗层网格到第一层网格逐层进行后光滑。

第 4 章三种常用网格的粗化方法

在实际应用中，我们在对 **Maxwell** 方程进行边型有限元离散得到有限元网格有下面 3 种：1.二维三角网格，2 三棱柱网格，3 三维四面体网格。下面介绍下 3 种网格的粗化方法。

4.1 二维三角网格粗化方法

对二维三角网格（见图 4.1）可直接用 RS 算法（见第 3 章）。

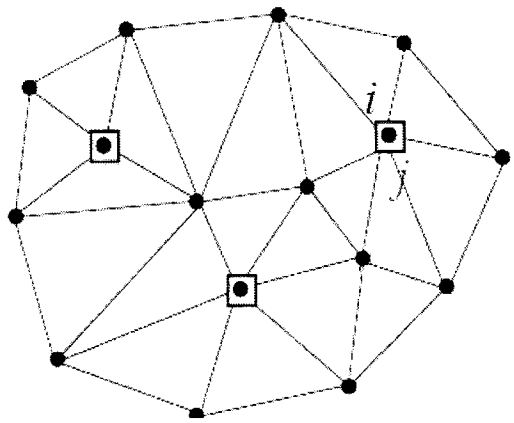


图 4.1　二维三角网格

4.2 三棱柱网格粗化算法

先介绍下三棱柱网格的几何特性：

1）三维网格由一定数量（一般小于 25）的完全相同的平面三角网格组成，并且每张平面网格的几何结构完全一样；

2）每两张平面网格的节点对之间有垂直边想连。

这样的网格可以看成是很多个三棱柱组成，见图 4.2。

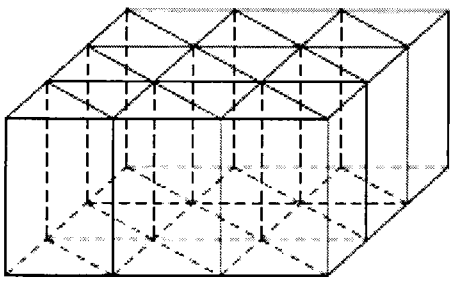


图 4.2 三棱柱网格

RS 粗化算法不能直接应用于三棱柱网格，因为 RS 粗化会在垂直方向上对点进行删选。而这样做可能会引起网络结构的混乱（粗化后得到的不再是三棱柱网格了），并且还会使得传递算子的定义变的复杂。我们根据三棱柱网格的几何特性设计更为合理的粗化算法。提取三棱柱网格 $w=(w^n,w^e)$ 的第一水平层三角网格，记为 $\tilde{w}=(\tilde{w}^n,\tilde{w}^e)$，例如，图 4.3 网格为从图 4.2 网格中提取出的水平层网格。

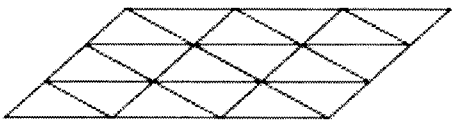


图 4.3 三维网格提取的水平层网格

提取的水平网格即是一个二维三角网格。则可对 $\tilde{w}$ 进行 RS 粗化，这个过程采用算法 3.2。令 $\widetilde{w_1}=\tilde{w}$，在对第 l 层有限元网格 $\widetilde{w_l}=(\widetilde{w_l}^n,\widetilde{w_l}^e)$ 粗化后，将 C 中点设为第 $l+1$ 层网格节点集 $\widetilde{\widetilde{w_{l+1}}}^n$，这样我们可以按照和 RS 粗化中完全一样的方法定一映射 $\widetilde{idx}:\widetilde{w_l}^n\to\widetilde{\widetilde{w_{l+1}}}^n$。

图 4.4 粗化后水平层网格

其中正方形点表示粗节点

辅助矩阵及其传递算子

粗化在一水平层网格上进行，因此从 l 层到 $l+1$ 层的辅助矩阵传递算子

$P_l^n \in \mathbb{R}^{\widetilde{N_l}^n \times \widetilde{N_{l+1}}^n}$ ($\widetilde{N_l}^n = |\widetilde{w_l}^n|, \widetilde{N_{l+1}}^n = |\widetilde{w_{l+1}}^n|$) 采取和（3.2）一样的定义：

$$(P_l^n)_{ij} = \begin{cases} 1 & j = \widetilde{idx}(i), i \in w_l^n, j \in w_{l+1}^n \\ 0 & else \end{cases} \tag{4.1}$$

其中 $\widetilde{idx}$ 为上节定义的从细节点 $\widetilde{w_l}^n$ 到粗节点 $\widetilde{w_{l+1}}^n$ 映射。

粗网格辅助矩阵根据 **Galerkin** 方法计算，即

$$B_{l+1} = (P_l^n)^T B_l P_l^n \tag{4.2}$$

B_{l+1} 代表第 $l+1$ 层虚拟有限元网格且 B_{l+1} 是 Z 矩阵。计算得辅助矩阵后，得到第 $l+1$ 层虚拟有限元网格 $\widetilde{w_{l+1}} = (\widetilde{w_{l+1}}^n, \widetilde{w_{l+1}}^e)$，见图 4.5。

图 4.5　水平层虚拟粗网格

系数矩阵及传递算子

辅助矩阵及起传递算子与网格节点对应，由于粗化只是在三棱柱网格的一个水平层网格进行，这样使得粗化和辅助矩阵的计算量比一般的代数多重网格法少好几倍（水平层网格个数）。

系数矩阵其传递算子是与边对应的，每两个水平层网格之间还有垂直边，因此传递算子的定义需在整个三棱柱网格上进行。

我们将水平层的粗化结构推广到整个三棱柱网格的粗化，即其它水平层网格和第一水平层网格采取一样的粗化（图 4.6），各水平层粗网格结构也是和第一层粗网格结构一样，两个水平层网格之间的粗网格节点构成粗边，这样得到粗化后的虚拟粗网格（图 4.7）。

接下来将$\widetilde{idx}:\widetilde{w_l}^n \to \widetilde{w_{l+1}}^n$沿拓到$idx:w_l^n \to w_{l+1}^n$，其中$w_l^n$为第$l$层三棱柱网格节点集合。我们定义$w_l^n$中节点按照水平层网络依次排列，而且不同水平层网格节点在本层中相对位置一样，$idx(\cdot)$可以通过下式得到：

$$idx(n)=(k-1)*\widetilde{N_{l+1}}^n+idx(\tilde{n}),n=k*\widetilde{N_l}^n+\tilde{n},0\le\tilde{n}<\widetilde{N_l}^n,0\le n\le N_l^n \quad (4.3)$$

其中$N_l^n=|w_l^n|$为第l层三棱柱网格点数，$\widetilde{N_l}^n=|\widetilde{w_l}^n|$为第$l$层水平层网格点数。

我们再定义w_l^e中边也是按照水平层网格依次排列，并且不同水平层网格边在本层中相对位置是一样，则从第l层到第$l+1$层的系数矩阵传递算子$P_l^e\in\mathbb{R}^{N_l^e\times N_{l+1}^e}(N_l^e=|w_l^e|,N_{l+1}^e=|w_{l+1}^e|)$，对于细网格边$i=(i_1,i_2)\in w_l^e,i_1<i_2$和粗网格边$j=(j_1,j_2)\in w_{l+1}^e,j_1<j_2,P_l^e$定义如下：

$$(P_l^e)_{ij}=\begin{cases}1 & (j_1,j_2)=(idx(i_1),idx(i_2))\\ -1 & (j_1,j_2)=(idx(i_2),idx(i_1))\\ 0 & else\end{cases} \quad (4.4)$$

我们根据三棱柱网格特性，对(4.4)进行分类讨论：

情况 1，如果i,j为同一个水平层网格上的粗边和细边采用（4.4）；

情况 2，如果i,j为连接两个水平层网格的细边和粗边即：

$i\in\widetilde{w_l}^n,i_2=i_1+\widetilde{N_l}^n\in\widetilde{w_l}^n,j_1\in\widetilde{w_{l+1}}^n,j_2=j_1+\widetilde{N_{l+1}}^n\in\widetilde{w_{l+1}}^n$，则采用下面定义

$$(P_l^e)_{ij}=\begin{cases}1 & j_1=idx(i_1)\\ 0 & else\end{cases}$$

情况 3，如果i,j为不同水平层网格上的细边和粗边，或者i,j为一条水平层网格上的边，一条为垂直方向的边则$(P_l^e)_{ij}=0$。

粗网格系数矩阵根据 **Galerkin** 方法计算即

$$A_{l+1}=(P_l^e)^T A_l P_l^e$$

系数矩阵具有 RS-AMG 系数矩阵的所有性质，见文献[16]。

4.3 三维四面体网格粗化方法

三维四面体网格是将上面的三棱柱网格的每个三棱柱分成三个四面体，这样的话 2 个平面间除了对应的点有边相连还有不对应的点有边相连，所以不能在一个面上粗化然后推广到其他面上去。因为代数多重网格是将网格完全数值化，我们还是采用原有的 RS 算法，对整个网格采用 RS 算法粗化。当然计算量随着辅助矩阵的规模的变大而提高。

第5章 以AMG做为预处理步骤的Krylov子空间预处理方法

作为一般的预处理Krylov子空间法，必需给出具体的预优矩阵M，而以**AMG**作为预处理步骤就是在预处理**Krylov**方法中出现的含有预优算子的方程组(如算法2.3.2中的$Mv = Ap$)用**AMG**算法求解，在求解过程中并不需要一个实际的M，而是用系数矩阵A近似的代替M，这样使预处理方法简便很多。

下面是**AMG**预处理的**BICGSTAB**法，参见文献[19]。

给定初值 x

$r = b - Ax$

$r = M^{-1} r_0$

$\rho = \alpha = w = \widetilde{\rho} = 1$

$v = p = 0$

$for \quad k = 1, 2, \ldots KMAX$

$\rho = (r_0, r)$

$\beta = \frac{\rho}{\widetilde{\rho}} \cdot \frac{\alpha}{w}$

$\widetilde{\rho} = \rho$

$p = r + \beta (p - w \cdot v)$

$t = Ap$

用AMG法求解$Av = t$中的t

$\alpha = \rho / (r_0, r)$

$s = r - \alpha v$

$t = As$

用AMG法求解$Az = t$中的z

$w = (t, s) / (t, z)$

$x = x + \alpha p + ws$

$r = s - wz$

$if \quad ||Ax - b|| / ||b|| \leq \varepsilon$

$break;$

end

end

而 **QMR** 算法也属于 Krylov 子空间法，并比 **BICG** 有更好的收敛性和稳定性所以很自然想到把 **AMG** 和预处理 **QMR** 算法也结合起来。

下面我们把 **AMG** 与算法 2.3.3 结合提出 **AMG** 预处理 **QMR** 算法。

给点初值 x_0

$r_0 = b - Ax_0, r = M^{-1}(b - Ax_0), \rho_0 = \| r \|, v_1 = r / \rho_0$

$V_1 = v_1, W_1 = v_1, D_1 = W_1^T V_1, \rho_1 = \xi_1 = 1$

$d_0 = 0, d_1 = V_1 D_1^{-1} W_1^T, f_0 = 0, f_1 = W_1 D_1^{-T} V_1^T$

for $n = 1, 2, \ldots$

$t = Av_n$

用代数多重网格法求解 $Az = t$ 得 z

$\widetilde{v_{n+1}} = z - d_1 z - d_0 z$

$\widetilde{w_{n+1}} = A^T w_n - f_1 A^T w_n - f_0 A^T w_n$

$\rho_{n+1} = \| \widetilde{v_{n+1}} \| \quad \xi_{n+1} = \| \widetilde{w_{n+1}} \|$

if $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$ *break*

end

$v_{n+1} = \widetilde{v_{n+1}} / \rho_{n+1} \qquad w_{n+1} = \widetilde{w_{n+1}} / \xi_{n+1}$

$V_{n+1} = [V_n \; v_{n+1}] \quad W_{n+1} = [W_n \; w_{n+1}] \quad D_{n+1} = W_{n+1}^T V_{n+1}$

$d_0 = d_1 \quad d_1 = V_{n+1} D_{n+1}^{-1} W_{n+1}^T \quad f_0 = f_1 \quad f_1 = W_{n+1} D_{n+1}^{-T} V_{n+1}^T$

得出 $AV_n = V_n H_n + [0 \ldots 0 \; \widetilde{v_{n+1}}]$ 中的 H_n

$$H_n^{'} = \begin{bmatrix} H_n \\ \rho_{n+1} (e_n)^T \end{bmatrix}, e_n = [0, \ldots 0 \; 1]^T \in \mathbb{R}^n,$$

对 $H_n^{'}$ 进行 QR 分解得 $Q_n H_n^{'} = \begin{bmatrix} R_n \\ 0 \end{bmatrix}$

$t_n = [I_n \;\; 0] Q_n d_n$ 其中 $d_n = \rho_0 e_{n+1} \quad e_{n+1} = [0, 0 \ldots 1] \in \mathbb{R}^{n+1}$

$x_n = x_0 + V_n R_n^{-1} t_n$

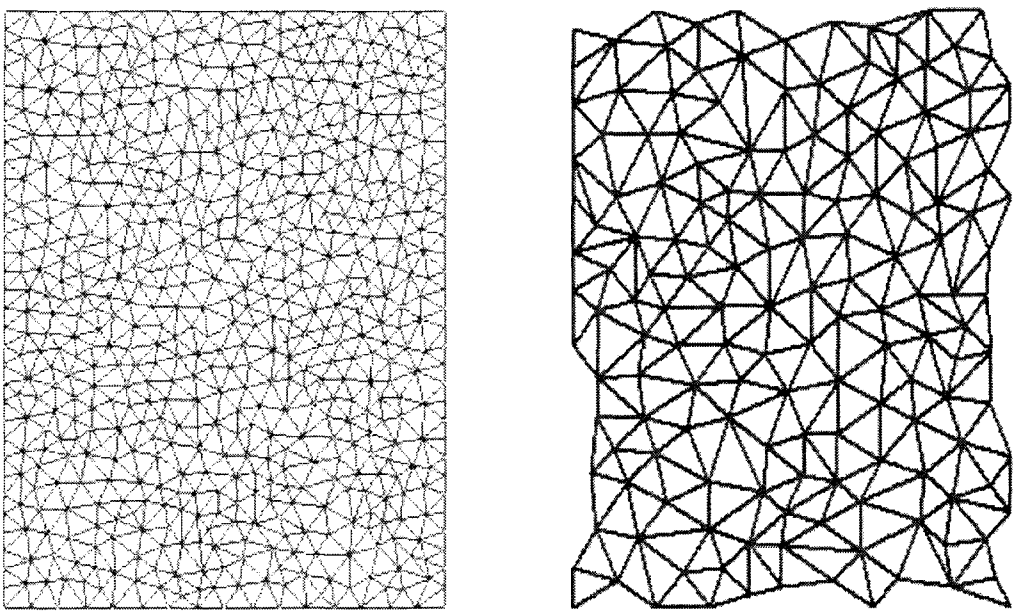
end

可行性分析：Osamu Tatebe 在[22]中详细讨论了多重网格法预处理矩阵 M 的性质，指出在满足一定条件下（比如用 **Damper Jacobi** 法或者红黑对称 Gauss-Seidel 法（**RB-SGC**）进行光滑），可以将多重网格法作为预处理法的预处理步骤具体见文献[22]或者[29]。

第6章　数值算例

下面我们先给出 2 个网格粗化的例子：

（1） 例 6.1 为一个平面有限元粗化结构，结构见图 6.1.。我们用第三章中的粗化算法对其进行粗化，得出下面的粗化结果



（a）细网格（627 个点，1782 条边）　　(b) 粗网格（159 个点，428 边）

图 6.1　一张平面有限元网格的粗化结果

（2） 例 6.2 为一张由 20664 个点，36800 个三棱柱构成的有限元网格，它由 21 张相同的平面网格连接而成。我们对其中一张平面网格（由 984 个点，2832 条组成）进行粗化，粗化后得到的最后一层平面网格又 6 个点，8 条边组成（与之对应的三棱柱网格由 126 个点，288 条边组成）粗化结果见图 6.2。

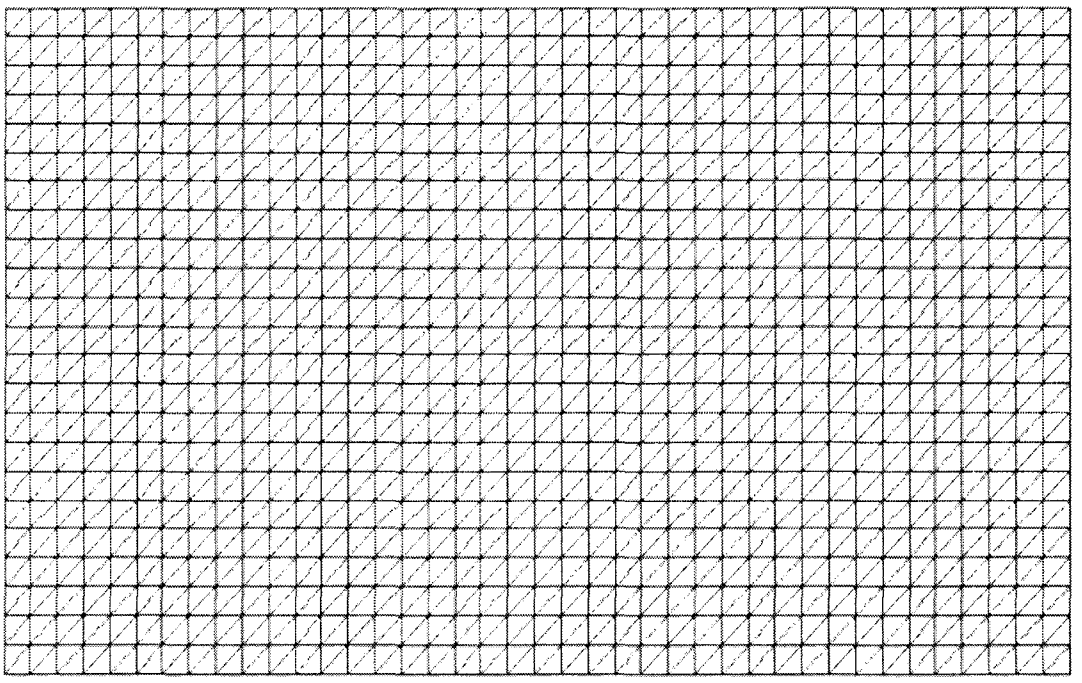


图 6.2.1 一张三棱柱有限元网格的平面网格粗化前结构

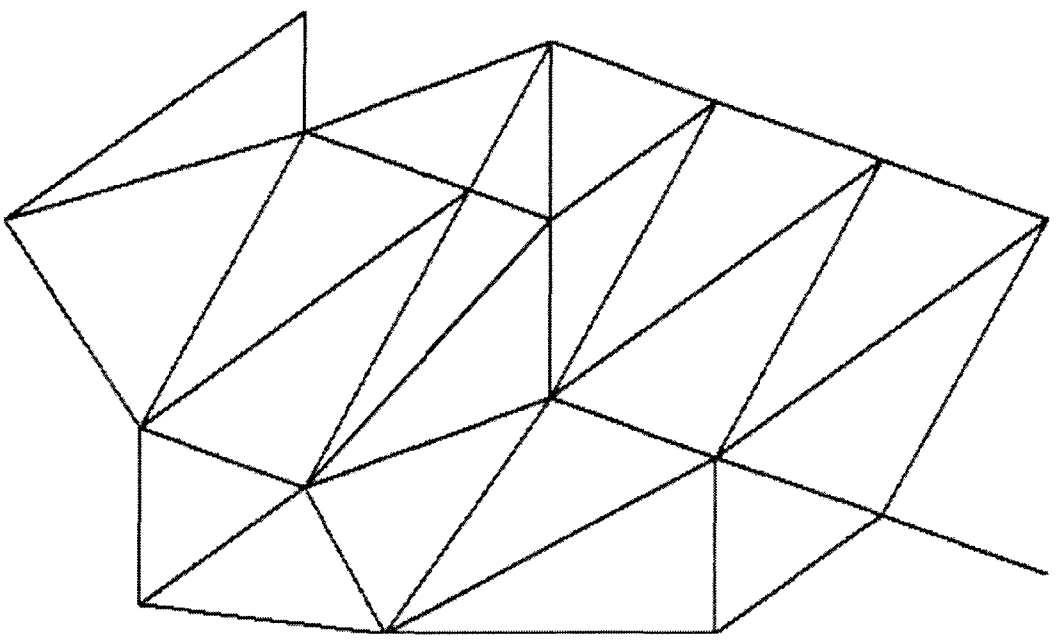


图 6.2.2 4 次粗化后结构，第 5 层网格，由 18 个点，37 条边组成

对应的第 5 层三棱柱网格又 378 个点，1137 条边组成

最后我们给出一个由 Maxwell 方程离散的例子，系数矩阵为复系数矩阵，有限元网格由 1131 个点 2000 个三棱柱组成，网格有 11 张平面网格连接而成。表 6.1 为粗化后线性方程组规模。

网格层数	未知数个数	系数矩阵非零元个数
1	4.730	48.796
2	2.315	23.401
3	652	6.090
4	213	1.876

表 6.1　粗化后线性方程组的规模

我们设定当残差$\|Ax-b\|_2/\|b\|_2$ $<1.0e-4$迭代终止，各迭代法收敛性见图 6.3。由于系数矩阵性质非常不好，GS（共轭梯度法）和 PCG（预处理共轭梯度法）不收敛，CCG，BICGSTAB 和 PBICGSTAB 以微弱速度收敛，而以 AMG 预处理 PQMR 法收敛速度提高很多。

图 6.3 横坐标为迭代次数，纵坐标为相对误差$\|Ax-b\|_2/\|b\|_2$

结束语

本文讨论了常用的代数多重网格法，并对常用的几种网格粗化做了下总结，最后把代数多重网格法和预处理 krylov 子空间法结合起来形成以代数多重网格为预处理算子的 krylov 子空间法，在求解大型稀疏矩阵方面有比较好的效率。

在以后的工作和研究中，AMG 算法的并行化尤其是粗化算法的并行化是研究的难点和重点。

参考文献

[1] Brandt A,McCormick S F,Ruge J,Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations[R].Institute for Computational Studies,Colorado,1982.

[2] Brandt A,Algebraic multigrid theory:the symmetric case[J].App Math Comp,1986,19:23-56.

[3] Concus P,Golub G H,O'Leary D P, A generalized conjugate gradient method for the numerical solution of elliptic patial diffierental equation. In:J.R.Bunch and D.J.Rose,eds.Sparse Matrix Computiongs,Academic Press,1976.309~332.

[4] Saad Y,GMRES:A Generalized minimal residual algorithm for solving nonsymmetric linear systems.SIAM J. Sci. Stat. Comput,1986.

[5] Saad Y,Wu K. dqgmres:A direct quasi-minimal residual algorithm based on incomplete orthogonalization:[Technical Report UMSI-93/131].Minnesota Supercomputing Institute,Minneanolis,MN,1993.

[6] Saad Y. Analysis of augmented krylov subspace techniques:[Technical Report UMSI-95/175].Minnesota Supercomputing Institute,1995.

[7] Sturler E. Nested krylov methods based on GCR. J. of computional and applied mathematics,1996.

[8] Sonneveld P, CGS: a fast lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput,1989,10:36~52.

[9] Fokkema D R,Sleijpen P,Van der vorst H A. Generalized conjugate gradient squared. J. Comput, Appl.Math,1996,71:125~146.

[10] Van der vorst H A. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems.SIAM J. Sci. Stat. Comput,1992,12:631~644.

[11] Freund R W, Nachtigal N M. QMR: A quasi-minimal residual method for non-hermitian linear systems,Numer. Math,1991,60:315~339.

[12] P.Joly, G.Meurant, Complex Conjugate Gradient Methods, Numerical. Algorithms, Volumn 4, Number 3, Oct 1993, 379-406, Springer Netherlands

[13] M.Jung, U. Langer, A. Meyer, W. Queck, M. Scheider, Multigrid preconditioners and their application, Proceedings of the 3rd GDR Multigrid Seminar held at Biesenthal, Karl-Weierstraβ-Institut für Mathematik, May 1989, 11-52.

[14] R. Barrett, M. Berry, T. Chan, J. Demmel , J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.

[15] J. W. Ruge, K. Stüben, Algebraic multigrid (AMG). In S.F. McCormick, editor, Multigrid Methods, volume 3 of Frontiers in Applied Mathematics, 73-130. SIAM, Philadelphia, PA, 1987.

[16] Stefan Reitzinger, Algebraic multigrid methods for large scale finite element equations – dissertation, Jan 2001.

[17] S. Reitzinger, J. Schöberl, Algebraic multigrid for edge elements. 15th Jun 2000.

[18] S. Reitzinger, J. Schöberl, An algebraic multigrid method for finite element discretizations with edge elements. Numer. Linear Algebra Appl., 9:223–238, 2002.

[19] 叶兴会. 代数多重网格法研究及其在信号完整性分析系统中的应用.浙江大学（硕士论文），2007.

[20] M. Jung, U. Langer, A. Meyer, W. Queck, M. Scheider, Multigrid preconditioners and their application, Proceedings of the 3rd GDR Multigrid Seminar held at Biesenthal, Karl-Weierstraβ-Institut für Mathematik, May 1989, 11-52.

[21] Hiptmair R. Multigrid method for Maxwell's equations. SIAM Journal on Numerical Analysis 1999; 36(1): 204-225.

[22] Osamu Tatebe, MGCG Method: A Robust and Highly Parallel Iterative Method. A Thesis Submitted to the Graduate School of the University of Tokyo, Partial Fulfillment of the Requirements for the Degree of Doctor of Science in Information Science, December 1996.

[23] Liu Qing-Huo, Weng Cho Chew, Applications of the conjugate gradient fast Fourier-Hankel transfer method with an improved fast Hankel transfor algorithm, Radio Science (ISSN 0048-6604), vol. 29, no. 4, p. 1009-1022.

[24] W. hackbusch, Multi-Grid Methods and Applications, Springer-Verlag Berlin,

Heidelberg, 1985.

[25] T. Grauschopf, M.Gribel, H. Regler, Additive multilevel preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs, SFB Bericht 342/02/96 A, technische Universität München, Institut für Informatik, 1996.

[26] M. Griebel, T. Neunhöffer, H. Regler, Algebraic multigrid methods for the solution of the Navier-Stokes equations in complicated geometries, Internat. J. Number. Methods Fluids 26(1998), no. 3, 281-301.

[27] M. Kaltenbacher, S. Reitzinger, J. Schöberl, Algebraic multigrid Method for Solving 3D Nonlinear Electrostatic and Magnetostatic Field Problems. Tech. Report, Johannes Kepler University Linz. SFB "Numerical and Symbolic Scientific Computing", 25 Oct. 1999.

[28] M. Adams, H. Bayrakrat, T. Keaveny, Applications Of Algebraic Multigrid to Large-Scale Finite Element Analysis of Whole Bone Micro-Mechanics on the IBM SP.

[29] A. Jemcov, J. P. Maruszewski, Stabilization and Acceleration of Algebraic Multigrid Method - Recursive Projection Algorithm, Oct 24, 20

致 谢

首先衷心感谢我的导师吴庆标教授，他有着严谨的治学态度，为人正直，知识渊博，在学习上给了我很大的帮助，并悉心的指导我，使我在硕士学习期间有了很大的进步，受益匪浅。除此之外，他还在生活上非常关心我，在我有困难的时候支持我，吴老师对我的影响和帮助是我一生巨大的财富。

感谢浙江大学计算数学方向的程晓良，黄正达老师，他们在教学中给了我很大的帮助，感谢我的师兄叶兴会和杭州迅美科技的张中庆教授，他们让我对代数多重网格方法从理论到实际应用有了一个比较全面的认识。

感谢身边的同学杨关祥，袁京欣，吴锋，陈记文，曾敏，杨森，孙隆刚，崔进等同学，他们在学习过程中给了我很大的帮助，并在其他方面也给了我很多宝贵建议。

此外还要感谢我的父母，妹妹，爷爷，奶奶，是他们在精神上和物质上给了我很大的支持，让我没有后顾之忧，可以勇往直前。

陈书浩

2008．5